МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра дискретной математики и информационных технологий

ВЫЧИСЛЕНИЕ ХАРАКТЕРИСТИК КОНЕЧНОГО СОСТОЯНИЯ В ЗАДАЧЕ О РОЖДЕНИИ ЧАСТИЦ ВНЕШНИМ ПОЛЕМ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ CUDA

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 271 группы
направления 09.04.01 Информатика и вычислительная техника
факультета компьютерных наук и информационных технологий
Окунева Сергея Олеговича

Научный руководитель:
доцент, к.ф.-м.н. _____ А.Д. Панферов

Зав. кафедрой:
доцент, к.ф.-м.н. _____ Л.Б. Тяпаев

Саратов 2019

ВВЕДЕНИЕ

Построение математических моделей является эффективным методом изучения различных явлений природы. Достаточно часто зависимость между различными характеристиками рассматриваемого физического, биологического или какого-нибудь другого динамического процесса не имеет прямого аналитического выражения а определяется только в форме дифференциальных уравнений, содержащих как текущие значения характеристик, так и их производные. Развитие вычислительной техники и алгоритмов решения дифференциальных уравнений различного вида сделало процедуры использования таких моделей достаточно обычной и успешной практикой. Вычислительная сложность реалистичных моделей многих процессов и явлений весьма высокой и быстро растет при стремлении повысить точность воспроизведения интересующих исследователя или инженера параметров. Эта проблема была и остаётся одним из главных стимулов развития вычислительных систем максимально высокой производительности.

В современных научных и технических задачах часто приходится сталкиваться с вычислительно сложными проблемами и алгоритмами. Для решения таких задач в разумные сроки современная компьютерная индустрия может предложить только путь распараллеливания и выполнения расчетов на массово-параллельных кластерных системах. В настоящее время в качестве лидирующей технологии, обеспечивающей лучшую эффективность, определилось использование специализированных вычислительных сопроцессоров на базе видеоускорителей. Именно с использованием такого подхода реализованы самые производительные вычислительные системы последнего поколения. Для написания параллельных версий программ для них существует специальный инструментарий, известный как технология CUDA.

Основной целью выпускной квалификационной работы является демонстрация результатов разработки решения для моделирования физических процессов с использованием этой технологии. Рассматривавшаяся модель описывает процессы, протекающие в новом, перспективном с точки зрения электроники материале, при действии на него внешних электрических полей. Такое моделирование является необходимым элементом разработки новых приборов и устройств с недостижимыми в настоящее время параметрами и характеристиками.

Проблема оказалась достаточно сложной. Хотя технологии CUDA уже более десяти лет, основной областью её использования является работа с большими векторами или матрицами. Практика многопотоковой реализации сложных вычислительных алгоритмов крайне ограничена. Тем не менее задача в её исходной постановке была решена и соответствующие результаты в работе представлены. Для этого были изучена и применена специализированная библиотека Thrust, предоставляющая высокоуровневые средства для работы со сложными структурами данных при обмене ими между памятью основного хоста и внутренней памятью видеоускорителя. Собственно процедуры решения системы ОДУ были реализованы с использованием библиотеки Boost.

Демонстрация работы разработанного решения выполнена на примере расчета отклика графена на действие короткого высокочастотного импульса с параметрами, соответствующими реальным экспериментам.

1 КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Модель поведения графена во внешнем электрическом поле» посвящен описанию моделируемой задачи.

Графен – это слой углерода толщиной в один атом, впервые полученный еще в 2004 году. Это самый прочный из всех материалов, известных на сегодняшний день, он очень гибкий и обладает высокой проводимостью. Данный материал обладает уникальными свойствами. Электроны в графене имеют постоянную скорость движения, которая в 300 раз меньше скорости света и называется скоростью Ферми.

Необычный отклик этого материала на появление внешнего электрического поля обуславливается сочетанием отсутствия массы и очень высокой скорости движения. Небольшого количества электронов с высокой скоростью движения появляется даже при слабом поле.

Используемая в работе расчетная модель должна воспроизводить эти особенности поведения материала и предсказывать новые для новых сочетаний параметров и условий.

1.1 Процедура получения (вычисления) данных

Подраздел посвящен описанию модели процессов, происходящих в графене под действием внешнего электрического поля. В нем представлены формулы и описания параметров.

Второй раздел «Понятие ОДУ и методы их решения» посвящен описанию обыкновенных дифференциальных уравнений и методам их решения.

2.1 Метод Эйлера

Метод Эйлера является простейшим численным методом решения систем обыкновенных дифференциальных уравнений.

Метод Эйлера является явным, одношаговым методом первого порядка точности, основанном на аппроксимации интегральной кривой кусочно-линейной функцией, так называемой ломаной Эйлера.

2.2 Методы Рунге-Кутты

Методы Рунге- Кутты — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем.

Методы Рунге-Кутты являются модифицированными методами Эйлера. Они служат для численного решения обыкновенных дифференциальных уравнений и их систем.

2.3 Метод Рунге – Кутты 4-го порядка

Наиболее часто реализуется и используется метод Рунге-Кутты четвертого порядка точности. В подразделе описываются формулы вычисления этого метода.

Третий раздел «Инструменты решения» посвящен описанию технологий и библиотек, использовавшихся в решении поставленной задачи.

GPGPU в переводе с английского означает General-purpose graphics processing units «GPU общего назначения» — техника для общих вычислений, обычно выполняющиеся на центральном процессоре, с использованием графического процессора видеокарты. То есть это набор программных и аппаратных технологий, который использует графические процессоры, для ускорения не графических приложений.

Для выявления основных различий между архитектурами графического и центрального процессоров необходимо отметить, что главной задачей центрального процессора является поочередное исполнение только одного потока команд с наибольшей производительностью, напротив конструкция графического процессора предполагает выполнение максимального числа потоков параллельно.

3.1 Технология CUDA

Технология CUDA — это программно-аппаратная вычислительная архитектура Nvidia, основанная на расширении языка Си, которая даёт возможность организации доступа к набору инструкций графического ускорителя и управления его памятью при организации параллельных вычислений. CUDA помогает реализовывать алгоритмы, выполнимые на

графических процессорах видеоускорителей Geforce восьмого поколения и старше.

3.1.1 Возможности Nvidia CUDA

Сложность разработки с использованием CUDA сильно зависит от самого алгоритма. Существуют алгоритмы, которые в принципе невозможно реализовать с использованием данной технологии. Такие программы нужно разделять между несколькими мультипроцессорами также как в MPI программировании, но без распределения данных, которые находятся в общей памяти на видеокарте. Но несмотря на все сложности программирования GPU с использованием технологии CUDA, трудозатраты все равно ниже, чем программирование с использованием ранних GPGPU решений.

3.1.2 Преимущества и ограничения CUDA

Программно-аппаратная модель для вычислений на GPU от компании Nvidia различается с предыдущими моделями GPGPU тем, что дает возможность написания программ для графических процессоров на языке C используя стандартный синтаксис, указатели и нуждается в минимуме расширений для использования вычислительных ресурсов видеочипов. CUDA обладает определенными особенностями, разработанными специально для вычислений общего назначения, а также она не зависит от графических API.

3.1.3 История развития CUDA

В подразделе описана хронология развития данной технологии, и основные отличия версий.

3.1.4 Состав Nvidia CUDA

В CUDA включены два вида API

- CUDA Runtime API (интерфейс высокого уровня);
- CUDA Driver API (интерфейс низкого уровня).

Нужно выбрать один из двух, так как использовать оба в одной программе невозможно. Высокоуровневый работает с использованием низкоуровневого, все вызовы Runtime API преобразуются в простые инструкции, обрабатываемые низкоуровневым Driver API.

3.1.5 Архитектура Nvidia CUDA

CUDA использует параллельную модель вычислений, когда каждый из процессоров выполняет одну и ту же инструкцию над различными элементами параллельно. GPU является вычислительным устройством, сопроцессором (device) для центрального процессора (host). Он обладает собственной памятью и параллельно обрабатывает большое количество потоков. Ядром (kernel) называется функция для GPU, которую исполняет каждый поток.

3.1.6 Модель памяти CUDA

Модель памяти в CUDA отличается возможностью побайтной адресации, поддержкой как gather, так и scatter. Доступно довольно большое количество регистров на каждый потоковый процессор. Доступ к ним очень быстрый, хранить в них можно 32-битные целые или числа с плавающей точкой.

3.1.7 Программирование CUDA

Программирование с использованием технологии CUDA отличается от традиционных API тем, что позволяет программисту писать на привычном C, только с небольшими расширениями.

Каждое ядро запускается в отдельном потоке. За формирование и компиляцию ядра отвечает центральный процессор. Видеочип копирует уже скомпилированный код для каждого элемента.

3.2 Библиотека Thrust

Библиотека Thrust является библиотекой шаблонов C++ для CUDA, аналогичной STL, но пока более простой. Она содержит только один вид контейнеров — векторы.

С помощью Thrust описываются вычисления, а не то, как будут храниться данные или производиться эти вычисления. В библиотеке обеспечивается абстрактный интерфейс к фундаментальным параллельным алгоритмам.

3.2.1 Векторы

Подобно вектору в STL — векторы в Thrust это общие контейнеры, хранящие любые типы данных и избавляющие программиста от забот с выделением/освобождением памяти, а также упрощающие обмен данными между CPU и GPU.

3.2.2 Алгоритмы

Thrust предоставляет большое количество алгоритмов многие из которых имеют аналог в STL, например *thrust::sort* и *std::sort*. Все алгоритмы имеют реализации как для хоста, так и для устройства. Нужный алгоритм выбирается в зависимости от входящих данных. Если входящие параметры будут принадлежать разным источникам, компилятор выдаст предупреждение об ошибке. Исключением является *thrust::copy*, который может копировать данные между хостом и устройством.

3.2.3 Итераторы

Так же как и в STL, thrust использует итераторы для задания места (или диапазона) в векторе (или массиве). Все контейнеры предоставляют стандартные методы, такие как *begin ()* и *end ()* для получения итераторов.

3.2.4 Пример программы

В подразделе рассмотрен пример применения библиотеки Thrust, выполняющий перемножение значения двух векторов и записывающей результат в третий вектор.

3.3 Библиотека Boost

Boost — собрание библиотек классов, использующих функциональность языка C++ и предоставляющих удобный кроссплатформенный высокоуровневый интерфейс для лаконичного кодирования различных повседневных подзадач программирования (работа с

данными, алгоритмами, файлами, потоками и т. п.). Свободно распространяются по лицензии Boost Software License вместе с исходным кодом. В данном разделе рассмотрена вторая версия библиотеки odeint v2 входящая в состав библиотеки Boost, отвечающая за решение обыкновенных дифференциальных уравнений.

3.3.1 Структура библиотеки

Библиотека состоит из трех основных частей:

- функции интегрирования;
- шаговые функции интегрирования (Stepper);
- вспомогательные функции.

3.3.2 Функции интегрирования

В параметры функции интегрирования передается некоторый stepper позволяющий решать ОДУ. Функции интегрирования вызывает методы *do_step* или *try_step* в зависимости от переданной пошаговой функции.

3.3.3 Пошаговые функции интегрирования

Stepper в odeint выполняет один единственный шаг. В odeint существует несколько разных типов пошаговых функций интегрирования. Они реализует концепцию Stepper, ErrorStepper, ControlledStepper, DenseOutputStepper.

Четвертый раздел «Практическая часть»

Для практической демонстрации изученного материала мне была поставлена задача написать программу, вычисляющую ансамбль систем обыкновенных дифференциальных уравнений. Для демонстрации основных возможностей решения задач с использованием технологии CUDA на современных графических сопроцессорах.

На первом этапе, в ходе изучения инструментов в рамках работы над практической частью была написана программа вычисляющую задачу Коши для ОДУ 1 порядка методом Рунге – Кутты 4 порядка. Программа написана на диалекте языке C++ в среде VISUAL STUDIO 2017 COMMUNITY. Целью

этого этапа была проверка возможности решения алгоритмически сложных задач использую технологию CUDA.

Но поскольку для реализации конечной цели возможно потребуются различные методы решения дифференциальных уравнений, желательно иметь возможность выбора между методами решения. В связи с этим при дальнейшей работе над проектом была поставлена задача решить уравнение не только методом Рунге – Кутты 4 порядка, но и другими методами, позволяющими, к примеру, контролировать размер шага и его погрешность. Реализация всех подобных методов «вручную с нуля» или переработка и оптимизация существующих библиотек, таких как GSL, Intel и пр. для технологии CUDA очень трудоемкая задача, потому было принято решение искать другие варианты реализации. В ходе поисков найдена библиотека с открытым исходным кодом Boost и пример программы, рассчитывающей экспоненту Ляпунова в системе Лоренца. Программа была написана с использованием библиотеки Thrust и Boost. Thrust представляет высокоуровневую абстракцию над технологией CUDA. Библиотека Boost предоставляет различные методы для решения дифференциальных уравнений.

На втором этапе написана программа на базе примера исходного кода, вычисляющая ансамбль систем Лоренца.

А также при помощи этих библиотек написана программа, решающая систему дифференциальных уравнений вида:

$$\begin{cases} f_1 = 0.5 * \ Lambda(t) * f_2(t) \\ f_2 = Lambda(t) * (1 - 2 * f_1) - 2 * ePsilon(t) * f_3 \\ f_3 = 2 * ePsilon(t) * f_2 \\ a_1 = -E_1(t) \\ a_2 = -E_2(t) \end{cases}$$

Ансамбль решается с различными входными параметрами**.**

Работа приложений была протестирована на ПК с видеокартой NVIDIA GeForce GT 640 а также на кластере высокопроизводительных вычислений ПРЦ НИТ СГУ.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы изучена технология реализации параллельных высокопроизводительных вычислений на видеоускорителях (CUDA). Для реализации процедур работы со сложными структурами данных в системах с неоднородным доступом к памяти (CPU + GPU) изучена библиотека Thrust. В качестве инструмента для решения обыкновенных дифференциальных уравнений изучены решатели библиотеки Boost, реализующие различные алгоритмы и процедуры их применения.

В результате это позволило разработать версию программного решения для моделирования поведения графена во внешнем поле через решение кинетического уравнения для носителей заряда, умеющую использовать высокий вычислительный потенциал современных видеоускорителей.

Работа программы успешно продемонстрирована на примере моделирования реальных экспериментов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Методы решения обыкновенных дифференциальных уравнений. [Электронный ресурс]: URL: https://www.swsu.ru/structura/up/fiu/tmim/Lec-NumMet.pdf (дата обращения: 25.05.2018) Загл. с экрана. Яз.рус.

2 Решение систем дифференциальных уравнений с помощью CUDA [Электронный Ресурс] URL: https://engraver.wordpress.com/2010/10/22/ode-solving-by-cuda/ (дата обращения 29.05.2019) Загл. с экрана Яз.рус.

3 CUDA Toolcit Documentation[Электронный Ресурс] URL: https://docs.nvidia.com/cuda/thrust/ (дата обращения 29.05.2019) Загл. с экрана Яз.англ.

4 DOCME [Электронный Ресурс] URL: https://www.docme.ru/doc/204964/nvidia-cuda--metoda--v1.2 (дата обращения 7.06.2019) Загл. с экрана Яз.рус.

5 Thrust [Электронный Ресурс] URL: http://thrust.github.io/ (дата обращения 29.05.2019) Загл. с экрана Яз.англ.

6 GitHub [Электронный Ресурс] URL: https://github.com/thrust/thrust/blob/master/examples/arbitrary_transformation.cu (дата обращения 29.05.2019) Загл. с экрана Яз.англ.

7 Code Project odeint v2 [Электронный Ресурс] URL: https://www.codeproject.com/Articles/268589/odeint-v2-Solving-ordinary-differential-equations (дата обращения 29.05.2019) Загл. с экрана Яз.англ.

8 Odeint solving ODEs in C++ examples [Электронный Ресурс] URL: http://headmyshoulder.github.io/odeint-v2/examples.html (дата обращения 29.05.2019) Загл. с экрана Яз.англ.

9 Steps3D [Электронный Ресурс] URL: http://steps3d.narod.ru/tutorials/thrust-tutorial.html (дата обращения 29.05.2019) Загл. с экрана Яз.рус.

10 Thrust [Электронный Ресурс] URL: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxjdWRhY3Ntc3VzdXxneDozYjZmMTQ0ZTRlYWIyOTYw (дата обращения 29.05.2019) Загл. с экрана Яз.рус.